Standards need to follow:

Keep in mind when you create/alter table.

1. Script should be written with if exists statement.
2. Column name must have proper data type and length, for example, DOB is required only date, use date data type instead of datetime.
3. Name must be specified for all type of constraints. (Foreign key, Default constraint, primary key, check constraints etc. Do not consider default name for any key/constraints.
4. If you think alter schema of any table, make sure all tables need to consider for alter which have similar/same column.
5. Data cleanup script must be there in foreign key scripts.
6. To create default constraints, think about exiting data.

Keep in mind when you create stored procedure

1. Use stored procedures instead of heavy-duty queries.
2. Include the SET NOCOUNT ON statement into your stored procedures to stop the message indicating the number of rows affected by a Transact-SQL statement.
3. Include SET XACT_ABORT ON statement into your stored procedures
4. Call stored procedure using its fully qualified name.
5. Consider returning the integer value as an RETURN statement instead of an integer value as part of a record set.
6. Don't use the prefix "sp_" in the stored procedure name if you need to create a stored procedure to run in a database other than the master database.
7. Use the sp_executesql stored procedure instead of the EXECUTE statement
8. If you have a very large stored procedure, try to break down this stored procedure into several sub-procedures, and call them from a controlling stored procedure.
9. Try to avoid using temporary tables inside your stored procedure. Please take approval from your lead to use temporary table.
10. Error handing should be part of stored procedure, it should be compulsory for those SPs who are having transactions.
11. Try to avoid Cursor in stored procedure, again take approval from lead.
12. Try to avoid using DDL (Data Definition Language) statements inside your stored procedure.
13. After writing stored procedure, please check execution plan; Execution plan should not have table scan option.

Keep in mind when you create SQL

1. Use EXISTS instead of IN to check existence of data.
2. Use EXISTS over COUNT(*) to verify data existence.
3. Proper use of Top Clause, see example below
4. Avoid * in SELECT statement. Give the name of columns which you require.
5. Choose appropriate Data Type. E.g. To store strings use varchar in place of text data type. Use text data type, whenever you need to store large data (more than 8000 characters).
6. Avoid nchar and nvarchar if possible since both the data type's takes just double memory as char and varchar.
7. Avoid NULL in fixed-length field. In case of requirement of NULL, use variable-length (varchar) field that takes less space for NULL.

8. Avoid Having Clause. Having clause is required if you further wish to filter the result of an aggregations.
9. Create Clustered and Non-Clustered Indexes.
10. Keep clustered index small since the fields used in clustered index may also used in non-clustered index.
11. Most selective columns should be placed leftmost in the key of a non-clustered index.
12. Better to create indexes on columns that have integer values instead of characters. Integer values use less overhead than character values.
13. Use joins instead of sub-queries.
14. Use WHERE expressions to limit the size of result tables that are created with joins.
15. Use UNION ALL in place of UNION if possible.
16. Use Schema name before SQL objects name.
17. Keep transaction as small as possible since transaction lock the processing tables data and may results into deadlocks.
18. Avoid prefix "sp_" with user defined stored procedure name because SQL server first search the user defined procedure in the master database and after that in the current session database.
19. Avoid use of Non-correlated Scalar Sub Query. Use this query as a separate query instead of part of the main query and store the output in a variable, which can be referred to in the main query or later part of the batch.
20. Avoid Multi-statement Table Valued Functions (TVFs). Multi-statement TVFs are more costly than inline TVFs.
21. After writing Query, Please check execution plan, Execution plan should not have table scan option.
22. When you see distinct or top1 in select query, please check twice whether it is table design issue or query issue.

----------SAMPLE QUERIES for performance----------

Choose highlighted one

```
SELECT  a.AddressLine1,
        a.AddressLine2,
        a.City,
        a.StateProvinceID
FROM    Person.Address AS a
WHERE   LEFT(a.AddressLine1, 4) = '4444';

SELECT  a.AddressLine1,
        a.AddressLine2,
        a.City,
        a.StateProvinceID
FROM    Person.Address AS a
WHERE   a.AddressLine1 LIKE '4444%' ;
```

Problem with Top Clause

Have you ever come across a situation where a SELECT query with a TOP clause will perform well most of the time, but as soon as you change the TOP value the same query is 10 to 20 times slower?

Solution: Let's take below example

```
--Source code provided by: www.sqlworkshops.com
SET NOCOUNT ON
CREATE TABLE tab7 (c1 INT PRIMARY KEY CLUSTERED, c2 INT, c3 CHAR(2000))
GO
```

```
BEGIN TRAN
GO
DECLARE @i INT
SET @i=1
WHILE @i<=50000
BEGIN
INSERT INTO tab7 VALUES (@i,RAND()*200000,'a')
SET @i=@i+1
END
COMMIT TRAN
GO
```

Update the statistic with a full scan to make the optimizer work easier.

```
UPDATE STATISTICS tab7 WITH fullscan
```

Let's set statistics running below query

```
SET STATISTICS time ON
GO
--Source code provided by: www.sqlworkshops.com
SELECT num_of_reads, num_of_bytes_read,
num_of_writes, num_of_bytes_written
FROM sys.dm_io_virtual_file_stats(DB_ID('tempdb'), 1)
GO
SELECT TOP 100 c1, c2,c3
FROM tab7
WHERE c1<30000
ORDER BY c2
GO
SELECT num_of_reads, num_of_bytes_read,
num_of_writes, num_of_bytes_written
FROM sys.dm_io_virtual_file_stats(DB_ID('tempdb'), 1)
```

```
Now SELECT num_of_reads, num_of_bytes_read,num_of_writes, num_of_bytes_written
FROM sys.dm_io_virtual_file_stats(DB_ID('tempdb'), 1)
GO
SELECT TOP 101 c1, c2,c3
FROM tab7
WHERE c1<30000
ORDER BY c2
GO
SELECT num_of_reads, num_of_bytes_read,num_of_writes, num_of_bytes_written
FROM sys.dm_io_virtual_file_stats(DB_ID('tempdb'), 1)
GO
```

So below query running to much slower than previous one

USE below logic

```
SELECT num_of_reads, num_of_bytes_read, num_of_writes, num_of_bytes_written
FROM sys.dm_io_virtual_file_stats(DB_ID('tempdb'), 1)
GO
DECLARE @i INT
SET @i=101
SELECT TOP(@i) c1, c2,CONVERT(VARCHAR(5000),c3)
FROM tab7
WHERE c1<30000
ORDER BY c2
GO
SELECT num_of_reads, num_of_bytes_read,num_of_writes, num_of_bytes_written
FROM sys.dm_io_virtual_file_stats(DB_ID('tempdb'), 1)
GO
```

Now the stats are exactly same as first query.
I think it is doing in memory operation for second query and not using tempdb. To verify this, execute below query; which indicates whether tempdb is use or not.
If any of the counters increase then tempdb is used...If no counters of below query increase then it is doing in memory operation...Hence it might be faster!

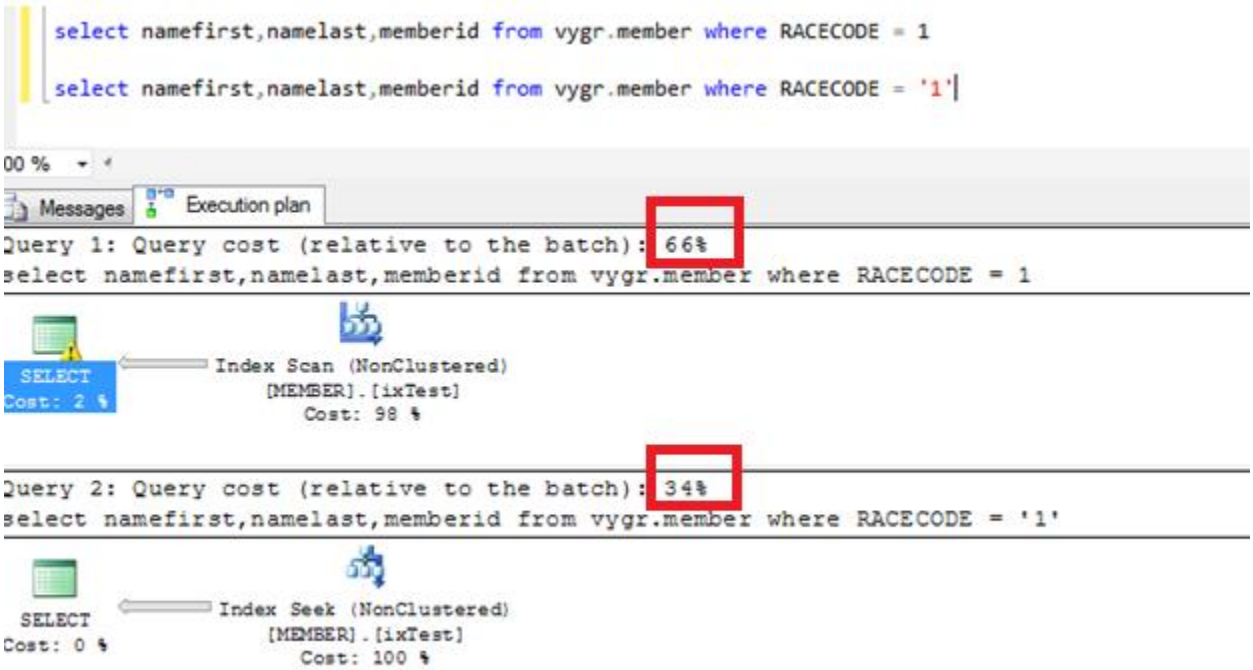
```
SELECT num_of_reads, num_of_bytes_read,
num_of_writes, num_of_bytes_written
FROM sys.dm_io_virtual_file_stats(DB_ID('tempdb'), 1)
```

Avoid convert to remove time from datetime

General process: `select convert(datetime,convert(varchar(10),getdate(),101))`

Recommended: `select dateadd(day,-datediff(day,getdate(),'1900-01-01'),'1900-01-01')`

Avoid implicit conversion:
Below queries result are same, but in performance too much difference. Always use explicit convert instead of implicit.



---

1. Use EXISTS over COUNT(*) to verify data existence.
2. Use UNION ALL over UNION.
3. Use indexes for aggregate and sort operations.
4. Avoid local variables in a batch query, if you have no option to avoid variable in batch query then use **option (optimize for variable),** see below.

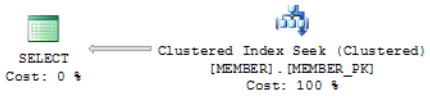
```
declare @min int
declare @max int
set @min = 10
set @max = 100
select * from vygr.member  where memberid > @min and  MEMBERID <=@max
select * from vygr.member  where memberid > 10 and  MEMBERID <=100
select * from vygr.member  where memberid > @min and  MEMBERID <=@max
OPTION (OPTIMIZE FOR (@min = 10, @max = 10))
```

100 %

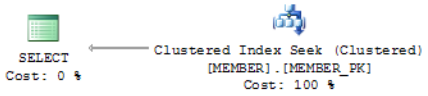
Messages | Execution plan

ASSIGN

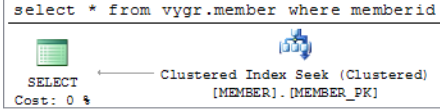
Query 3: Query cost (relative to the batch): 84%
select * from vygr.member where memberid > @min and MEMBERID <=@max



Query 4: Query cost (relative to the batch): 13%
select * from vygr.member where memberid > 10 and MEMBERID <=100



Query 5: Query cost (relative to the batch): 3%
select * from vygr.member where memberid > @min and MEMBERID <=@max OPTION (OPTIMIZE FOR (@min = 10, @max = 10))



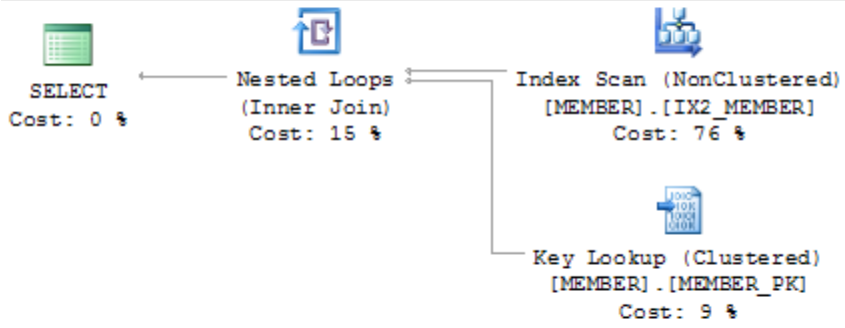
Avoid arithmetic operator in where clause

```
select * from vygr.member where (memberid*2) = 200
select * from vygr.member where memberid = (200/2)
```

100 %

Messages | Execution plan

Query 1: Query cost (relative to the batch): 92%
select * from vygr.member where (memberid*2) = 200



Query 2: Query cost (relative to the batch): 8%
select * from vygr.member where memberid = (200/2)